Docket No.: M4065.0433/P433
Micron Ref. No.:00-0897

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR U.S. LETTERS PATENT

Title:

METHOD AND CIRCUIT FOR NORMALIZATION OF FLOATING POINT SIGNIFICANDS IN A SIMD ARRAY MPP

Inventor:

Graham Kirsch

DICKSTEIN SHAPIRO MORIN &
OSHINSKY LLP
2101 L Street NW
Washington, DC 20037-1526
(202) 785-9700

METHOD AND CIRCUIT FOR NORMALIZATION OF FLOATING POINT SIGNIFICANDS IN A SIMD ARRAY MPP

FIELD OF THE INVENTION

[0001] The present invention relates to the field of massively parallel processing systems, and more particularly to a method and apparatus for efficiently normalizing and aligning the significand portion of a floating point number in a single instruction multi data massively parallel processing system.

BACKGROUND OF THE INVENTION

[0002] The following application is related to application serial number 09/_____ filed on _______, entitled "Method and Circuit for Alignment of Floating Point Significands in a SIMD Array MPP," the disclosure of which is incorporated by reference.

[0003] The fundamental architecture used by all personal computers (PCs) and workstations is generally known as the von Neumann architecture, illustrated in block diagram form in Fig. 1. In the von Neumann architecture, a main central processing unit (CPU) 10 is coupled via a system bus 11 to a memory 12. The memory 12, referred to herein as "main memory", also contains the data on which the CPU 10 operates. In modern computer systems, a hierarchy of cache memories is usually built into the system to reduce the amount of traffic between the CPU 10 and the main memory 12.

[0004] The von Neumann approach is adequate for low to medium performance applications, particularly when some system functions can be accelerated by special purpose hardware (e.g., 3D graphics accelerator, digital signal processor (DSP), video encoder or decoder, audio or music processor, etc.). However, the approach of adding accelerator hardware is limited by the bandwidth of the link from the CPU/memory part of the system to the accelerator. The approach may be further limited if the bandwidth is shared by more than one accelerator. Thus, the processing demands of large data sets are not served well by the von Neumann architecture. Similarly, as the processing becomes more complex and the data larger, the processing demands may not be met even with the conventional accelerator approach.

[0005] Referring now to Fig. 2, an alternative to the von Neumann architecture is the single instruction multiple data (SIMD) massively parallel processor (MPP) system. A MPP system differs from a von Neumann system by using a large number of processors, called processing elements (PE) 200, coupled to a communications network 15. The communications network 15 permit each PE 200 to exchange data with other PEs 200. Additionally, the PEs 200 may read or write to main memory 12 via an array-to-memory bus 13, or receive commands or instructions from CPU 10 via bus 11. Although the CPU 10 may perform some processing, in a SIMD MPP system, the array of PEs 14, comprising the PEs 200 and its communications network 15, perform most of the computations. The CPU 10 functions in a supporting role.

[0006] In a SIMD MPP, each PE operates on the same instruction, at the same time, but on different pieces of data. Since the PEs in a SIMD array operate in lockstep, data dependent conditional operations cannot be performed by branching, as would be done in a conventional processor. Instead, each PE can decide whether to store the result of an operation either in an internal register or in a memory dependent upon a condition generated within the PE from data local to the PE. This technique is known as "activity control" and is a very powerful method for performing data dependent decisions in a parallel computer which operates on a single stream of instructions.

[0007] Most SIMD MPPs utilize relatively simple processors for PEs 200. For example, short integer PEs 200, such as 8-bit integer processors may be used. SIMD MPPs utilize these simple processors in order to increase the number of PEs 200 which can be integrated upon a single silicon die. High performance is achieved by the use of a large number of simple PEs 200, each operating at a high clock speed.

[0008] The use of short integer PEs 200 mean that floating point operations may require several clock cycles to complete. In many computer systems, floating point numbers are often stored in a manner consistent with the IEEE-754 standard. In particular, the IEEE-754 standard stores single precision floating point number as three binary fields taking the format of:

$$(-1)^s \times 2^{(e-127)} \times (1.f) \qquad (1)$$

wherein:

s is a single bit representing the sign of the floating point number.

e is an 8-bit unsigned integer representing a biased exponent. e is said to represent a biased exponent because the actual exponent being represented is equal to e - 127. Although an 8-bit unsigned integer may range from 0-255, and thereby permitting exponents in the range from -127 (i.e., -127 = 0 - 127) to +128 (i.e., 128 = 255 - 127), the IEEE-754 standard limits the range of usable exponents to exclude -127 and +128.

1.f is a 24-bit significand field in a "normalized" format, i.e., a bit field in which the most significant bit (MSB) is the first digit left of the binary point and in which the most significant bit is set to one. Since the most significant bit of a normalized number is understood to be 1, there is no need to store the most significant bit.

[0009] Data which have biased exponents of 0 and 255 are used to represent special conditions and the number zero. The IEEE-754 standard represents the number zero using a biased exponent of 0 (i.e., for the single precision format, the exponent equals

-127) and a significand field of 00000000000000000000000$_2$. (In the special cases of zero and non-normalized numbers, indicated by the exponent being 0, the most significant bit of the significand is not taken to be a 1.)

[0010] Under the IEEE-754 standard, single extended, double, and double extended precision numbers are stored in similar format, albeit using different sized exponents and significands. For example, double precision numbers use a 10-bit biased exponent field with representable exponents ranging from -1022 to 1023 and a significand having 53 bits.

[0011] In order to perform arithmetic operations on floating point number stored in the IEEE-754 format, the floating point numbers first need to be separated, or "demerged", to extract the sign bit, the exponent, and the significand. Once these fields have been extracted, they can be operated upon in order to perform the arithmetic operation. For example, multiplying two floating point number includes multiplying the significands and adding the exponents. Once the arithmetic operation has been performed, significand field of the result may not be in a normalized format. For example, multiplication of two operands with normalized significands results in an answer ranging from 0_2 to 100_2. The process of returning a significand field back to a normalized format is known as normalization.

[0012] In conventional computer systems, normalization is normally performed using standard shifting logic, such as barrel shifters. Shifting logic is used in conventional computer systems because they have adequate speed and they do not consume a significant amount of silicon real estate in comparison to the other circuitry in a complex CPU 10. However, in a SIMD MPP using simple PEs 200, standard shifting logic such as barrel shifters would significantly increase the size of the PEs 200 and also be too slow. Accordingly, there is a desire and need for a way to efficiently perform normalization of floating point significands in a SIMD MPP environment.

SUMMARY OF THE INVENTION

[0013] The present invention is directed at a processing element of a SIMD MPP which can efficiently perform the normalization processes commonly used when performing arithmetic operations on floating point numbers. The PEs of the SIMD MPP include two groups of registers. One of the groups is known as the M block and includes a plurality of registers and logic which permits limited right shifting of the contents of the registers. The other group of registers is known as the Q block and includes a plurality of registers and logic which permits limited left shifting (e.g., 1-, 2-, 4-, and 8- bit left shifts are supported) of the contents of the registers. A method is used with the limited left shifting ability of the Q block registers to normalize the result of an arithmetic calculation.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The foregoing and other advantages and features of the invention will become more apparent from the detailed description of the preferred embodiments of the invention given below with reference to the accompanying drawings in which:

[0015] FIG. 1 is a block diagram of a prior art von Neumann architecture computer system;

[0016] FIG. 2 is a block diagram of a SIMD MPP computer system;

[0017] FIG. 3 is a block diagram of one of the PEs in the SIMD MPP computer system in accordance with the principles of the present invention;

[0018] FIGS. 4A and 4B are a flow chart which illustrate how the PE of the present invention aligns significand data; and

[0019] FIG. 5 is a flowchart which illustrates how the PE of the present invention normalizes significand data.

DETAILED DESCRIPTION OF THE INVENTION

[0020] Now referring to the drawings, where like reference numerals designate like elements, there is shown in Fig. 3 a block diagram of a PE 200 in accordance with the principles of the present invention. The PE 200 is divided into several functional blocks, including an ALU 301, which is coupled to a Node Communications Interface 305 and a DRAM Interface 303. The Node Communications Interface 305 is used by the PE 200 to send and receive messages to the four other PE 200 adjacent to the present PE 200, over signal lines 306a, 306b, 306c, and 306d. The DRAM Interface 303 is used by the PE 200 to read and write to a main memory 12. The ALU 301 is also coupled to a series of registers, including a register file 302 used to store data, a series of flag registers 307, and a shift control register ("SCR") 360. In the exemplary embodiment, the SCR 360 is an 8-bit register with the most significant bit designated bit 7 and the least significant bit designated bit 0. The function of the flag registers 307 and the SCR 360 will be explained later. The PE 200 also includes two registers blocks, namely the M Block 350a and the Q Block 350b.

[0021] The M block 350a includes a bus called the M Bus 307a which is coupled to the Node Communications Interface 305. The M bus 307a is also coupled, via logic circuit 308a to a plurality of registers. These registers include the M3 310, M2 311, M1 312, M0 313, and MS 314 registers. In some embodiments an optional a G register 320 may also be present. The G register 320 may be used, for example, to store extension bits for use in higher precision calculations. In one exemplary embodiment, registers M3, 310, M2, 311, M1 312, and M0 313 are 8-bit registers while register MS 314 is a single bit register. Logic circuit 308b couples registers M3 310, M2 311, M1 312, M0 313, MS 314, and G 320 to Q Bus 307b, ALU 301 and DRAM Interface

304. The logic circuits 308a and 308b represent conventional logic circuits such as a network of multiplexers, which permit the registers M3 310, M2 311, M1 312, M0 313, MS 314, and G 320 to receive and transmit data in a manner which will be described in additional detail.

[0022] Additionally, logic circuits 308a, 308b are also capable of demerging an IEEE-754 formatted number into its sign, biased exponent, and significand fields. In particular, the sign is stored in register MS 314, the biased exponent is stored in M3 310, and the significand is stored in registers M2 311 (most significant byte), M1 312, and M0 313 (least significant byte). The logic circuits 308a, 308b may also be capable of setting registers M2 311, M1 312, and M0 313 to zero. Finally, logic circuits 308a, 308b also permit data stored in registers M2 311 and M1 312 to be right shifted in increments of 1, 2, 4, and 8 bits. The M registers (i.e., MS 314, M0 313, M1 312, M2 311, and M3 310) and the Q registers (i.e., QS 334, Q0 333, Q1 332, Q2 331 , and Q3 330) are coupled via signal line 307c. This permits the contents of the M registers to be transferred in one clock cycle to corresponding Q registers in the Q block.

[0023] The Q block 350b is similar to the M block 350a. The Q block has an bus known as the Q bus 307b. The Q bus 307b is not coupled to the Node Communications Interface 305. Instead, the Q bus 307b is coupled via signal line 307c to the M Bus 307a of the M block 350a. The Q block 350b include a series of Q registers, namely QS 334, Q0 333, Q1 332, Q2 331, and Q3 330. In the exemplary embodiment register QS is a single bit register while registers Q0 333, Q1 332, Q2 331, and Q3 330 are 8-bit registers. The Q block 350b has logic circuits 309a, 309b which function in a manner similar to logic circuits 308a, 308b of the M block 350a. One significant difference between the two sets of logic circuits, 308a/308b and 309a/309b, however, is that while logic circuits 308a, 308b permit data stored in registers M2 and M1 to be right shifted in 1, 2, 4, and 8 bit increments, logic circuits 309a, 309b permit data in registers Q2 331 and Q1 332 to be left shifted, in the same increments.

[0024] The PE 200 also includes a flag register 307 which contain a plurality of flags. These flags default to being set to zero, unless a specific conditions resets them to one. In the exemplary embodiment there are four flags named Q2Z8, Q2Z4, Q2Z2, and Q2Z1, which function as described below. Flag Q2Z8 is one if all eight bits of register Q2 331 are zero. Flag Q2Z4 is one if the four most significant bits of register Q2 331 are zero. Flag Q2Z2 is one if the two most significant bits of register Q2 331 are both zero. Finally, flag Q2Z1 is one if the most significant bit of register Q2 331 is zero.

[0025] The PE 200 performs floating point arithmetic operations by first demerging the two IEEE-754 formatted operands. This is done by loading the first operand into the M block 350a. The operand may be loaded from the Node Communications Interface 305 if the operand is sent from an adjacent PE 200.

Alternatively, the operand may be loaded from the DRAM Interface 303 if the operand had been loaded into the main memory 12. As mentioned previously, the logic circuits 308a, 308b in M block 350a demerge an IEEE-754 formatted operand into its sign, biased exponent, and significand fields by storing the sign field in register MS 314, the biased exponent in register M3 310, and the significand in registers M2 311 and M1 312. Once the first operand has been demerged, it is transferred via signal line 307c to the Q block 350b. The second operand is then loaded to the M block 350a and demerged. At this point, the two demerged successive operands are in the M block 350a and the Q block 350b.

[0026] The ALU 301, which is coupled to the M block 350a via logic circuit 308b and the Q block 350b via logic circuit 309b, is used to perform the arithmetic operation in an ordinary manner. For example, the significands may be added, subtracted, or multiplied. For addition and subtraction the exponents of the operands are equal and do not require adjustment. For multiplication, the exponents are summed. The result of the arithmetic operation are stored in the Q block 350b. As usual, the most significant byte of the result is stored in register Q2, and lesser significant bytes of the results are progressively stored in registers Q1 and Q0. If there are additional bits of the result which needs storing, the lesser significant bytes of the results may be stored in the G register 320 (if present) and the M0 register 313 of the M Block 350, and additional lesser significant bytes of the results may be stored in the register file.

[0027] After performing the arithmetic operation, the significand may not be in normalized form. In order to comply with the IEEE-754 standard, the significand stored in the plurality of Q registers Q2 331 Q1 332 Q0 333 may need normalization. In general, the result of an arithmetic operation may result in a significand having a number of zeros (up to the level of precision, i.e., up to 24 for IEEE-754 single precision arithmetic) at the most significant portion of the significand. The normalization process shifts the significand so that the most significant bit (i.e., bit 7 of register Q2 331) is a one.

[0028] The normalization of the significand is performed according to the 7 steps described below and illustrated in Fig. 5, steps 500-515:

[0029] (Step 1) Set a temporary variable, such as one of the registers in the register file 302 to zero (Fig. 5, 501).

[0030] (Step 2) If flag Q2Z8 is equal to one (Fig. 5, 502), shift the result to the left by eight bits and add 8 to the temporary variable (Fig. 5, 503).

[0031] (Step 3) If flag Q2Z8 is equal to one (Fig. 5, 504), left shift the result by 8-bits and add 8 to the temporary variable (Fig. 5, 505).

[0032] (Step 4) If flag Q2Z8 is equal to one (Fig. 5, 506), left shift the result by 8-bits and add 8 to the temporary variable (Fig. 5, 507).

[0033] (Step 5) If flag Q2Z4 is equal to one (Fig. 5, 508), left shift the result by 4-bits and add 4 to the temporary variable (Fig. 5, 509).

[0034] (Step 6) If flag Q2Z2 is equal to one (Fig. 5, 510), left shift the result by 2-bits and add 2 to the temporary variable (Fig. 5, 511).

[0035] (Step 7) If flag Q2Z1 is equal to one (Fig. 5, 512), left shift the result by 1-bit and add 1 to the temporary variable (Fig. 5, 513).

[0036] (Step 8) The exponent of the result is adjusted by subtracting the temporary variable from the exponent. I.e., Q3 = Q3 - temporary variable (Fig. 5, 514).

[0037] Note that as the shifting is performed in the Q registers Q2 331 Q1 332 Q0 333, the contents of the G register 320 is being shifted into register Q0. Likewise the contents of the M0 313 register is being shifted into register G 320.

[0038] For example, suppose in one of the PEs 200 of the array 14, the Q Block 350b registers (Q3 330, Q2 331, Q1 332, and Q0 333) contain the following values:

Q3	Q2	Q1	Q2
0000 1000	0001 0101	1001 1001	0000 1111

[0039] Normalization is performed as follows: In step (1), a temporary variable is set to zero. The temporary variable may be a register from the register file 302, a memory location accessed via the DRAM Interface 304, or any other temporary storage location. The content of the registers, flags, and temporary variable after step (1) are as follows:

Q3	Q2	Q1	Q0
0000 1000	0001 0101	1001 1001	0000 1111

Q2Z8	Q2Z4	Q2Z2	Q2Z1	Temp
0	0	1	1	0

[0040] In step (2) since flag Q2Z8 is equal to zero so no further processing is performed in step (2). The content of the registers, flags, and temporary variable after step (2) are as follows:

Q3	Q2	Q1	Q0
0000 1000	0001 0101	1001 1001	0000 1111

Q2Z8	Q2Z4	Q2Z2	Q2Z1	Temp
0	0	1	1	0

[0041] In step (3) since flag Q2Z8 is equal to zero, no further processing is performed in step (3). The content of the registers, flags, and temporary variable after step (3) are as follows:

Q3	Q2	Q1	Q0
0000 1000	0001 0101	1001 1001	0000 1111

Q2Z8	Q2Z4	Q2Z2	Q2Z1	Temp
0	0	1	1	0

[0042] In step (4), since flag Q2Z8 is equal to zero, no further processing is performed in step (4). The content of the registers, flags, and temporary variable after step (4) are as follows:

Q3	Q2	Q1	Q0
0000 1000	0001 0101	1001 1001	0000 1111

Q2Z8	Q2Z4	Q2Z2	Q2Z1	Temp
0	0	1	1	0

[0043] In step (5), since flag Q2Z4 is equal to zero, no further processing is performed in step (5). The content of the registers, flags, and temporary variable after step (5) are as follows:

Q3	Q2	Q1	Q0
0000 1000	0001 0101	1001 1001	0000 1111

Q2Z8	Q2Z4	Q2Z2	Q2Z1	Temp
0	0	1	1	0

[0044] In step (6), since flag Q2Z2 is equal to one, the content of registers Q2, Q1, and Q0 are right shifted by 2-bits, and 2 is added to the temporary variable. The content of the registers, flags, and temporary variable after step (6) are as follows:

Q3	Q2	Q1	Q0
0000 1000	0101 0110	0110 0100	0011 1100

Q2Z8	Q2Z4	Q2Z2	Q2Z1	Temp
0	0	0	1	2

[0045] In step (7), since flag Q2Z1 is one, the content of registers Q2, Q1, and Q0 are right shifted by 1-bit, and 1 is added to the temporary variable. The content of the registers, flags, and temporary variable after step (7) are as follows:

Q3	Q2	Q1	Q0
0000 1000	1010 1100	1100 1000	0111 1000

Q2Z8	Q2Z4	Q2Z2	Q2Z1	Temp
0	0	0	0	3

[0046] In step (8), the contents of the temporary variable (now 3) is subtracted from the exponent (which is held in register Q3). The contents of the Q registers are now normalized and the state of the registers, flags, and temporary variable (at this point the temporary variable is no longer needed and may be used for other purposes) are as follows:

Q3	Q2	Q1	Q0
0000 0101	1010 1100	1100 1000	0111 1000

Q2Z8	Q2Z4	Q2Z2	Q2Z1	Temp
0	0	0	0	3

[0047] Thus, the present invention provides an apparatus and a method for normalizing the significand portion of an floating point number, such as those which follow the IEEE-754 floating point standard, in a SIMD MPP environment. The present invention is advantageous in that each PE 200 of the array 14 is not required to have a full feature shifter, such as a barrel shifter. Instead, a faster but more limited shifting logic, such as logic circuits 308a, 308b, which are only capable of shifting the significand data by 1-, 2-, 4-, or 8- bits are used in combination with a shift control register 360, under a nine step procedure to align the significand. Ideally, the instruction or instructions which correspond to each of the nine steps can be executed by a PE 200 in a single clock cycle. Since in a SIMD environment each PE 200 in the array 14 executes the same instruction at the same time, every significand in the array 14 can be aligned in as little as nine clock cycles.

[0048] Although the invention has been discussed and illustrated in the context of a 8-bit shift control register and shifting circuits which are capable of shifting significand data by 1-, 2-, 4-, and 8- bits, the invention is not so limited and may be generalized as follows: The flexibility of the left shifting circuitry and the number of flags may be varied. The number of flags and the flexibility of the left shifting circuitry is related as follows. If there are F+1 flags (wherein F is an integer of at least 3), then the left shifting circuitry should be capable of left shifting the significant being normalized by 2^0, 2^1, 2^2, …, or 2^F bits.

[0049] The generalized normalization procedure begins with the arithmetic logic unit setting to zero the value of a temporary storage location. Each flag is then examined, beginning with flag F and ending with flag 0. For each flag which is equal to one, the arithmetic logic unit causes the left shifting circuitry to left shift the significand by 2^F bits and add 2^F to the value stored in the temporary storage location. After every flag has been analyzed, the value stored in the temporary register is subtracted from the significand's exponent.

[0050] While certain embodiments of the invention have been described and illustrated above, the invention is not limited to these specific embodiments as numerous modifications, changes and substitutions of equivalent elements can be made without departing from the spirit and scope of the invention. Accordingly, the scope of the present invention is not to be considered as limited by the specifics of the particular structures which have been described and illustrated, but is only limited by the scope of the appended claims.